Lessons Learned

Lessons Learned: Bilko

> **Project:** Bilko — Balkan Accounting SaaS **Version:** 0.1 **Date:** 2026-02-23
> **Author:** John (AI Director) **Status:** Draft **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John (AI Director)	Initial entry — Phase 1 kickoff learnings

---

> "Lessons learned without behavior change is not learning." — Alem Bašić, 2026-02-13 (ZAKON #0)

Every entry in this log must produce either a systemic fix (hook/tool/rule) or a concrete behavior change. Observations without action items do not belong here.

---

1. Session Metadata — Phase 1 Kickoff Review

Field	Value
Session Type	Phase Review (after Pipeline Gate 8 pass + initial build)
Project	Bilko
Phase / Sprint	Pre-Sprint 1 — Phase 1 kickoff
Period Covered	2026-02-19 – 2026-02-23

Field	Value
Facilitator	John (AI Director)
Participants	John, Alem Bašić
Date of Session	2026-02-23
Duration	Retrospective of first 4 days

2. What Went Well (Keep Doing)

#	What Went Well	Why It Worked	How to Preserve	Category
KD-01	8-gate pipeline forced all critical research before coding began	Gates prevented building on wrong assumptions; regulatory research (MULTI-REGION-OVERVIEW.md) was detailed and accurate	Continue using pipeline gates for Phase 2 (Croatia) and Phase 3 (BiH); enforce gate passage before any new country's features	Process
KD-02	Database schema (15 models) designed before writing any application code	Schema aligned with accounting law requirements (double-entry, NUMERIC, audit trail) from day 1; no retroactive migrations	Always design data model first for financial features; NEVER write API code against an undefined schema	Technical
KD-03	NUMERIC(19,4) for all monetary fields decided upfront and enforced in schema	Prevented entire class of floating-point accounting errors; found in Fiken research	CI test: fail if any new field for monetary values uses Float/Decimal without explicit 4dp precision; document in CLAUDE.md	Technical
KD-04	Regulatory research completed and documented before building	SEF API requirements, PDV rates, Croatian eRačun timing — all known before design phase	Research-first mandate in project charter; Asmir as ongoing regulatory advisor	Process
KD-05	CEO approval gate (Gate 8) prevents building orphaned features	Alem reviewed full plan before code was written; GO decision is documented and traceable	Maintain Gate 8 for each major phase/country expansion	Process

Highlights

KD-02 Detail: The decision to define all 15 Prisma models (Organization, User, Invoice, InvoiceItem, Transaction, etc.) before writing a single API endpoint proved to be the most valuable architectural decision of Phase 1. When the backend starts being built, the developer (human or AI) has a complete, validated schema to work against — no guessing about data shapes. The schema covers double-entry (debitAccountId + creditAccountId in Transaction), multi-currency (ExchangeRate model with locked rates), and audit trail (LoggedAction append-only). This pattern should be standard for all future Bilko features.

---

3. What Didn't Go Well (Stop Doing)

#	What Didn't Work	Root Cause	Impact (1-10)	Action Item	Owner	Deadline
SD-01	Frontend built entirely with mock data — 8 pages, zero real API connections	Prototype-speed MVP prioritized demo-ability over production-readiness; no explicit decision to replace mock data before launch	10	CI check: grep for mock-data.ts imports outside test files; fail build in staging/production	John	Sprint 1
SD-02	packages/ui component library left empty — all components in apps/web, creating future duplication risk	"We'll extract later" reasoning; never happened during MVP build	4	Document as TD-003; schedule for Phase 2 before mobile work starts	John	2026-05-01
SD-03	No CI/CD pipeline established before writing application code	Infrastructure "can wait" assumption; now blocking safe iteration	8	CI/CD setup is Sprint 1 task 1 before any API endpoints	John	2026-03-07

#	What Didn't Work	Root Cause	Impact (1-10)	Action Item	Owner	Deadline
SD-04	API-REFERENCE.md not created alongside the schema and routes structure	Documentation always deferred in early phases; creates frontend/backend misalignment	5	Every new API endpoint must have its Request/Response schema documented in API-REFERENCE.md in the same PR	John	Ongoing
SD-05	No unit tests written during MVP build	Speed prioritized; tests "added later" — never happened	9	Test must be written in the same sprint as the feature, not deferred	John	Ongoing

Root Cause Analysis — Deep Dives

SD-01: Mock Data Throughout Frontend

Problem: All 8 frontend pages show hardcoded mock data. Beta testers would see fake numbers. Cannot test real API integration.

5 Whys:

1. Why is all data mocked? → Frontend was built as a prototype to show design/flow, not as production code
2. Why was it built as prototype only? → Gate 8 (CEO approval) was needed before real backend investment; prototype reduced risk
3. Why wasn't mock replacement scheduled explicitly? → The PIPELINE.md "Next Steps" listed it but no MC task was created with a deadline
4. Why was no MC task created? → Prototype phase didn't have sprint planning; items added to informal list
5. Why did informal list not get actioned? → No hook or gate enforcing the replacement before beta access

Root Cause: No automated enforcement that mock data must be replaced before staging deployment. **Systemic Fix:** CI check in `apps/api/src/` and `apps/web/` — grep for `mock-data.ts` imports outside `/test/` directories; fail build if found in staging/production environment. Add to `.github/workflows/ci.yml`. **Ljestvica Fikseva (ZAKON #1):** Hook > Tool > Rule > CLAUDE.md > Memory → Using CI hook (strongest enforcement)

SD-05: No Tests Written During MVP Build

Problem: 0% test coverage on financial logic (PDV, double-entry, SEF). Accounting software without tests is a liability — a PDV calculation bug could cause users to file incorrect tax returns.

5 Whys:

1. Why are there no tests? → They were "deferred" to after the feature was "working"
2. Why were they deferred? → Speed pressure to show CEO a working prototype
3. Why did "working" not include tests? → No definition of "done" included test coverage
4. Why was there no DoD? → Project was in prototype phase with informal standards
5. Why were informal standards accepted? → No enforcement mechanism (no CI, no PR gate)

Root Cause: No CI gate requiring tests before merge; no DoD including test coverage. **Systemic Fix:** GitHub Actions CI: `npm test -- --coverage --coverageThreshold='{"global":{"lines":80}}'` — fails PR if coverage below 80% overall or below 95% for `src/services/` (financial logic). This is non-negotiable before Sprint 2 merges. **Ljestvica Fikseva:** CI coverage gate → fails PR merge.

---

4. What to Try Next (Start Doing)

#	What to Try	Expected Benefit	Hypothesis	Success Metric	Trial Period
ST-01	Write tests FIRST (TDD) for financial logic (PDV, double-entry, SEF)	Catches accounting bugs before they reach users; forces clear spec before coding	If we write Given/When/Then tests for PDV calculation before writing the calculation, then PDV accuracy errors drop to 0	Zero PDV calculation bugs in beta; CI balance check passes 100%	Sprint 2-4
ST-02	Create API-REFERENCE.md endpoint-by-endpoint alongside backend implementation	Eliminates frontend/backend sync issues; enables frontend developer (or agent) to start immediately	If every endpoint has documented Request/Response before frontend implements it, then integration time drops 50%	Zero "unexpected API shape" issues during frontend-backend connection sprint	Sprint 2
ST-03	SEF sandbox test on every PR touching invoice logic	Catches SEF breaking changes before they reach production users	If we run SEF sandbox test in CI for every invoice change, then SEF regression bugs = 0 in production	Zero SEF regressions post-launch	Sprint 3 (when SEF built)

---

5. Key Findings by Category

5.1 Technical Findings

#	Finding	Impact	Recommendation
TF-01	Prisma's Decimal type maps cleanly to NUMERIC(19,4) — zero extra work for financial precision	High positive	Continue using Decimal for all monetary Prisma fields; never use Float
TF-02	LoggedAction as append-only audit table is a strong pattern — enforces immutability at schema level	High positive	Apply same pattern to other sensitive tables in future products (Drop, BasicFakta)
TF-03	Turborepo monorepo structure enables shared packages (database, ui) but requires discipline — empty packages create confusion	Medium	Either fill packages immediately or don't create them; no empty scaffolds
TF-04	Next.js 15 App Router + React Server Components significantly changes how data fetching works — team must understand before building	High	Document RSC vs client component decision boundary in CLAUDE.md

Technical Patterns Observed:

- "Schema first, endpoints second, frontend third" — the correct build order for financial software; schema ensures all features are feasible
- Monorepo shared packages need a charter: what goes in packages/ui, what stays in apps/web — decide before Sprint 2

5.2 Process Findings

#	Finding	Root Cause	Recommendation
PF-01	8-gate pipeline prevented scope creep effectively — every decision was documented before coding	Explicit gate criteria with documented evidence	Apply same pipeline to Croatia (Phase 2) and BiH (Phase 3); don't skip gates even if process feels familiar

#	Finding	Root Cause	Recommendation
PF-02	"Next Steps" in PIPELINE.md without MC tasks = invisible work	No translation from documentation to task management	Every "Next Step" in any doc must have a corresponding MC task created immediately
PF-03	Prototype → production transition requires explicit decision and enforcement, not assumption	Prototype mode is sticky — it continues until something forces a change	Launch gate checklist must include "zero mock data imports verified by CI"

Process Patterns Observed:

- Documentation-first works for planning but requires enforcement mechanisms to drive action
- Sprint planning discipline is critical even for AI-only teams — without sprints, work becomes unordered

5.3 Communication Findings

#	Finding	Stakeholder Impact	Recommendation
CF-01	Weekly Slack summary to Alem keeps CEO aligned without overhead meetings	Alem can make informed go/no-go decisions asynchronously	Maintain weekly sprint summary format from Communication Plan
CF-02	Regulatory questions need Asmir's input — cannot be resolved by research alone	Wrong SEF implementation would cause launch failure	Asmir review at every SEF-touching sprint; not just at start

5.4 Tools & Infrastructure Findings

#	Finding	Impact	Recommendation
IF-01	No CI/CD pipeline means every deploy is a manual risk	High — production incidents have no automated safety net	CI/CD is Sprint 1 task 1, not an afterthought
IF-02	SEF sandbox environment must be set up before SEF integration sprint begins	Medium — delay if sandbox access not ready	Alem/Asmir to secure SEF sandbox credentials by 2026-03-01

5.5 Team & Collaboration Findings

#	Finding	Impact	Recommendation

TM-01	AI agent team (John + builder/validator agents) works well for structured implementation tasks but requires extremely detailed specs — vague requirements produce vague code	High	Every FR must have explicit Acceptance Criteria before being assigned to a builder agent
TM-02	Financial domain requires accounting expertise that pure coding agents don't have — PDV, double-entry, SEF need validation by someone who knows the law	Critical	Asmir review for all regulatory features; automated accounting logic tests are the safety net

---

6. Action Items

#	Action Item	Category	Owner	Sprint / Deadline	Priority	Status
AI-01	Set up CI/CD pipeline (GitHub Actions) — include: lint, type-check, tests, coverage gate, mock-data import check	Infrastructure	John	Sprint 1 (by 2026-03-07)	P1	Open
AI-02	Write unit tests for tax.service.ts (PDV calculation) as first test file — TDD model	Testing	builder agent	Sprint 2 start (2026-03-07)	P1	Open
AI-03	Create `apps/api/docs/API-REFERENCE.md` — document every endpoint as it is built	Documentation	John	Ongoing from Sprint 1	P2	Open

#	Action Item	Category	Owner	Sprint / Deadline	Priority	Status
AI-04	Add CI check: grep for `mock-data.ts` in non-test files — fail build if found in staging env	Tools/Infrastructure	John	Sprint 1 (by 2026-03-07)	P1	Open
AI-05	Secure SEF sandbox credentials from APR via Asmir	Regulatory	Alem + Asmir	By 2026-03-01	P1	Open
AI-06	Create MC tasks for every "Next Step" in PIPELINE.md	Process	John	Now	P2	Open
AI-07	Asmir to review SEF integration design before Sprint 3 implementation	Regulatory	John	By 2026-03-14	P1	Open
AI-08	Define RSC vs client component decision boundary for Bilko in CLAUDE.md	Technical	John	Sprint 1	P2	Open

---

7. Metrics Comparison — Planned vs. Actual

7.1 Timeline (Phase 1 kickoff only)

Milestone	Planned	Actual	Variance	Root Cause
All 8 pipeline gates	2026-02-19	2026-02-20	+1 day	Gate 5-8 validation took longer than expected; thorough — good

Milestone	Planned	Actual	Variance	Root Cause
Backend foundation (auth, 4 endpoints)	2026-02-20	2026-02-20	0	On target
Full 50 endpoints	2026-03-07	TBD	TBD	Sprint 1

7.2 Budget (Phase 1 so far)

Category	Planned (EUR)	Actual (EUR)	Variance	Notes
Research + design + schema	~€1,500	~€800	-€700	AI-assisted research more efficient than estimated
Backend foundation (auth)	~€500	~€300	-€200	4 endpoints in 1 session
Running total	**~€2,000**	**~€1,100**	**-€900**	Under budget so far

7.3 Quality

Metric	Target	Actual	Status
Test coverage	≥ 80%	0% (not yet written)	In progress
Critical bugs at this stage	0	0	
Pipeline gate pass rate	8/8	8/8	
Database schema completeness	15 models	15 models	

---

8. Recommendations for Future Projects

#	Recommendation	Category	Confidence	Applicable When
REC-01	Define all database models before writing any API endpoints or frontend code	Technical	High	Any data-driven application; especially financial
REC-02	Use pipeline gates (8-gate or adapted) before investing in build phase for any new product	Process	High	All new ALAI products

#	Recommendation	Category	Confidence	Applicable When
REC-03	Set up CI/CD in Day 1 of Sprint 1 — never defer infrastructure; it becomes load-bearing	Technical	High	Every project
REC-04	For regulated domains (accounting, finance, health): TDD is not optional; tests are the compliance proof	Technical	High	Any compliance-regulated feature
REC-05	Research regulatory requirements with a local expert (Asmir for Balkans) before designing features that touch them	Process	High	Any Balkan market expansion

Top 3 Most Important Learnings

1. **Schema First, Always:** The 15-model Prisma schema designed before writing a single line of application code is why Bilko's data architecture is sound. The schema was validated against accounting law (double-entry), PRD features, and multi-currency requirements all at once. Future features should follow the same discipline: design the data model, get it reviewed, then build.
2. **Mock Data Is a Prototype Trap:** Shipping a frontend with mock data creates a false sense of completeness. The 8 pages look done but nothing works end-to-end. In future builds: either build frontend and backend in parallel (stub real API from day 1), or explicitly schedule "replace mock with real" as a first-class sprint task with CI enforcement.
3. **Regulatory Compliance Is Not a Sprint Task — It's an Ongoing Advisor Relationship:** SEF is not just "call an API." It's a living government system with evolving requirements, certificate requirements, and edge cases that only practitioners know. Asmir's involvement is not optional — it's risk mitigation. Budget time for regulatory review at every sprint touching financial compliance.

---

9. Knowledge Base Contribution

Finding	Contribution Type	Target System	Status	Owner
NUMERIC(19,4) for all monetary fields	Rule added to Bilko CLAUDE.md	`~/ALAI/products/Bilko/CLAUDE.md`	Done (already in CLAUDE.md)	John

Finding	Contribution Type	Target System	Status	Owner
Pipeline gate process	Template	`~/ALAI/products/Bilko/PIPELINE.md`	Done	John
Mock data replacement pattern	HiveMind entry	HiveMind	Open	John
Schema-first pattern	HiveMind entry	HiveMind	Open	John

HiveMind entries to add:

```
node ~/system/agents/hivemind/hivemind.js post "Bilko pattern: Always design Prisma schema before API endpoints. Schema = contract. Validated against accounting law + PRD features before writing any code."
node ~/system/agents/hivemind/hivemind.js post "Bilko lesson: Mock data in frontend is a prototype trap. Enforce removal with CI grep check before staging deployment. See CROSS-CUTTING/tech-debt-log.md TD-001."
```

Approval

Role	Name	Date	Signature
Author	John (AI Director)	2026-02-23	
Reviewer			
Project Manager	John	2026-02-23	
AI Director (John)	John	2026-02-23	
Team	Alem Bašić		

Revision #3
Created 2026-02-24 23:11:28 UTC by John
Updated 2026-05-31 20:04:23 UTC by John